Programmation linéaire en nombres entiers pour la recherche d'isomorphisme de sous-graphe : Application à la recherche de symboles graphiques

Pierre Le Bodic, Pierre Héroux, Sébastien Adam, Hervé Locteau, Jean Noel Bilong Mboumba, Yves Lecourtier

Programmation linéaire en nombres entiers pour la recherche d'isomorphisme de sous-graphe

Application à la recherche de symboles graphiques

Pierre Le Bodic[*] — Pierre Héroux[**] — Sébastien Adam[**] — Hervé Locteau[***] — Jean-Noël Bilong Mboumba[**] — Yves Lecourtier[**]

** Univesité de Paris-Sud - LRI UMR 8623 , F-91405 Orsay cedex*

*** Université de Rouen - LITIS , F-76800 Saint-Etienne du Rouvray*

**** LORIA UMR 7503, BP239 - F-54506 Vandoeuvre les Nancy*

RÉSUMÉ. *Cet article propose une approche basée sur la programmation linéaire en nombres entiers pour la recherche d'isomorphisme de sous-graphes. L'originalité de l'approche réside dans la prise en compte d'un étiquetage du graphe par des vecteurs numériques pour lesquels les méthodes traditionnelles sont inopérantes. Cette approche est appliquée à la localisation de symboles dans des documents graphiques décrits par graphes d'adjacence de régions étiquetés par descripteurs de formes et des caractéristiques exprimant des relations géométriques. La globalité de l'approche est évaluée sur une base de documents synthétiques. Les résultats obtenus sont prometteurs.*

ABSTRACT. *This paper proposes an approach based on integer linear programming to solve the subgraph isomorphism problem. This approach is able to handle graphs labeled with numerical feature vectors for which traditional methods can not be used. This approach is applied for symbol localization in technical document images decribed with region adjacency graphs labeled with shape descriptors and feature vectors describing geometric relations. The whole approach is evaluated on a synthetic document database. Preliminary results are encouraging.*

MOTS-CLÉS : *Détection de symboles, Isomorphisme de sous-graphes, Programmation linéaire en nombres entiers*

KEYWORDS: *Symbol spotting, subgraph isomorphism, integer linear programming*

1. Introduction

Les graphes étiquetés sont des structures de données appréciées pour leur aptitude à représenter des entités complexes. Au sein d'une représentation à base de graphe, les nœuds et leurs étiquettes décrivent des objets ou des parties tandis que les arcs représentent les relations entretenues entre les objets. En raison de la généricité intrinsèque des représentations à base de graphes, et grâce à l'augmentation de la puissance de calcul des ordinateurs, les représentations structurelles sont devenues de plus en plus utilisées dans de nombreux domaines d'application tels que la biologie, la chimie, l'analyse d'images de document ou la reconnaissance de symboles. Une conséquence de l'usage intensif des représentations à base de graphes est un intérêt croissant pour les problématiques scientifiques associées que sont la fouille de graphes, la classification de graphes (supervisée ou non) ou la recherche d'isomorphisme.

Cet article aborde le problème de la recherche d'isomorphisme de sous-graphe et propose une application dans le cadre de la détection de symboles dans des documents graphiques. En exploitant des graphes de régions adjacentes attribués pour représenter les documents et les symboles, nous présentons un nouveau cadre pour la recherche d'isomorphisme de sous-graphe afin de trouver les instances de symboles dans les documents. Ce nouveau cadre modélise l'isomorphisme de sous-graphe comme un problème d'optimisation formulé comme un Programme Linéaire en Nombres Entiers (PLNE). La formulation proposée vise à trouver une correspondance exacte du point de vue de la topologie du graphe, tout en tolérant les erreurs du point de vue des étiquettes des nœuds et des arcs. Le problème est résolu par les méthodes d'optimisation combinatoire implémentées par des solveurs tels que, par exemple, SYMPHONY[1] (Ralphs *et al.,* 2005). Le système de recherche de symboles, dans sa globalité, est évalué sur un ensemble de 200 documents synthétiques contenant 5609 occurrences de symboles provenant de 16 classes. Ces documents sont produits par l'application décrite par Delalandre *et al.* dans (Delalandre *et al.,* 2008).

De nombreux approches ont été proposées pour la localisation de motifs dans une image. Certaines agissant au niveau pixelaire et se basant sur la recherche de zones caractéristiques (SIFT(Lowe, 1999), RIFT(Lazbenik *et al.,* 2004), SURF(Bay *et al.,* 2008)) peuvent trouver une application pour la recherche de symboles et offrir de très bons résultats. Nous ne prétendons pas concurrencer de telles approches. Notre contribution pricipale réside plutôt dans la proposition d'une méthode de recherche d'isomorphisme de sous-graphe apte à traiter des graphes étiquetés numériquement et pour laquelle la localisation de symbole n'offre qu'un cadre applicatif adapté où les représentations structurelles sont fréquemment utilisées.

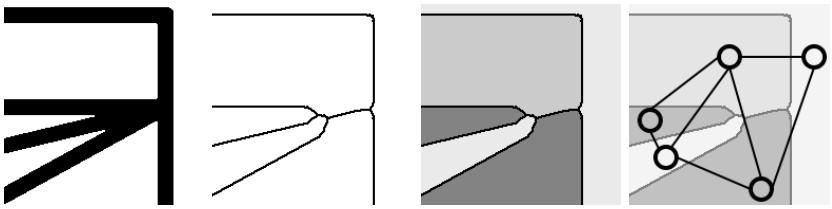
1. `http ://www.coin-or.org/SYMPHONY/`

Le reste de cet article est organisé de la façon suivante. En section 2, nous présentons succinctement le processus permettant d'extraire les représentations structurelles des images de documents et de symboles. La section 3 présente la modélisation du problème de recherche d'isomorphisme par un programme linéaire en nombres entiers. En section 4, nous décrivons le protocole expérimental, les bases de données utilisées et les résultats obtenus lors de l'évaluation de l'approche, avant de dresser en section 5 les conclusions et de présenter quelques voies permettant de poursuivre ce travail.

2. Représentations structurelles

La détection de symboles est un des problèmes relevant de l'analyse d'image de document. Il s'agit de trouver l'emplacement de certains symboles dans une image de document. Ce type de problèmes revêt une difficulté supérieure à celui de la reconnaissance de symboles isolés dans le sens où il est nécessaire de simultanément segmenter et reconnaître le symbole. De fait, si la littérature abordant la reconnaissance de symboles est abondante, très peu d'approches sont proposées pour la détection de symboles (Queshri *et al.,* 2006, Rusiñol *et al.,* 2009). Dans cet article, nous présentons un système visant la détection de symboles dans un contexte particulier. Tout comme les travaux de Lladós *et al.* (Lladós *et al.,* 2001), l'approche proposée s'appuie sur deux niveaux. Le premier niveau permet l'extraction d'une représentation structurelle à base de graphes d'adjacence de régions. Cette représentation est extraite aussi bien de l'image de symbole à rechercher que sur l'image du document où sont cherchées les occurrences du symbole. Le second niveau vise à trouver les occurrences du graphe modèle représentant le symbole au sein du graphe représentant le document. Là où Lladós *et al.* proposent une description des régions par la chaine constituée frontières issues d'une étape de vectorisation sensible aux perturbations, nous proposons d'utiliser des descripteurs de formes. Par ailleurs, l'algorithme de recherche d'isomorphisme de sous-graphe utilisé dans (Lladós *et al.,* 2001) procède par agrégation successive de régions voisines, la rendant tout à fait adaptée à la recherche de symboles. La méthode que nous proposons, décrite en section 3 se veut moins contraignante et pouvant être utilisée dans d'autres contextes applicatifs.

Nous décrivons, dans cette section, le premier niveau de l'approche proposée. Les graphes d'adjacence de régions sont des structures de données permettant une bonne description des symboles graphiques dans le sens où ils autorisent la modélisation des relations topologiques entre les régions extraites grâce à un processus de segmentation. Nous traitons des images de documents techniques (images binaires) où la composante blanche est associée au fond tandis que les composantes noires correspondent à la partie graphique. La segmentation de telles images peut être obtenue par étiquetage des composantes (Chang *et al.,* 2004). Cependant, afin d'obtenir une représentation fine des relations



(a) Image originale (b) Squelette de l'image (a) (c) Composantes connexes de l'image (b) (d) Graphe d'adjacence de régions

Figure 1. *Création du graphe d'adjacence de régions*

d'adjacence pour chaque paire de régions, l'image binaire est soumise à une squelettisation (di Baja *et al.,* 1996). L'image obtenue est identique à l'image originale du point de vue morphologique, mais l'épaisseur de la composante graphique est réduite à 1 pixel. On fait alors correspondre à chaque composante blanche de cette image squelettisée un nœud dans le graphe en construction. Par ailleurs, un parcours des branches du squelette est exploité pour déterminer les relations d'adjacence entre les régions deux à deux. Cette relation d'adjacence est matérialisée par la création d'un arc entre les nœuds associés aux régions correspondantes. La figure 1 illustre, sur un extrait d'image de document, le processus de construction du graphe d'adjacence de régions.

Afin de caractériser les nœuds représentant les régions et de préciser la nature des relations d'adjacence, le graphe est étiqueté. Plusieurs types de caractéristiques ont été proposées dans la littérature pour décrire les formes et les relations spatiales (Terrades *et al.,* 2007). Parmi les descripteurs de formes, les moments de Zernike (Teague, 1980) permettent d'atteindre de bonnes performances lors de la reconnaissance de formes soumises à des transformations affines ou des dégradations. Un vecteur de caractéristiques composé des 24 premiers moments de Zernike extraits de chaque composantes connexes et caractérisant la forme est donc utilisé pour étiqueter les nœuds correspondants dans le graphe. Le graphe construit est dirigé et les attributs affectés aux arcs ($source \rightarrow destination$) sont :

– une caractéristique liée au rapport des surfaces des composantes associées aux nœuds source et destination ;

$$\sqrt{\frac{A(destination)}{A(destination) + A(source)}}$$

– une caractéristique liée à la distance entre les centres de gravité des régions associées aux nœuds origine et destination.

$$\frac{d_e(g_{source}, g_{destination})}{\sqrt{A(source) + A(destination)}}$$

Au final, on obtient un graphe $G = (V, E, L_v, L_E)$ où V et $E \subseteq V \times V$ sont respectivement les ensembles de nœuds représentant les régions et d'arcs représentant les relations d'adjacence entre les régions. L_v et L_E sont les fonctions d'étiquetage des nœuds et des arcs.

– $L_V : V \mapsto \mathbb{R}^d$, permet une description de la forme de la région associée.

– $L_E : E \mapsto \mathbb{R}^2$, permet une description de la relation d'adjacence.

Du fait de la réciprocité de la relation d'adjacence, l'existence d'un arc ij implique l'existence d'un arc ji. Si l'étiquetage de ces arcs est identique du point de vue de la caractéristique relative à la distance entre les centres de gravité des régions, chaque arc apporte une information différente sur la caractéristique relative au rapport des surfaces.

Un graphe d'adjacence de régions $\mathcal{G}$ est construit de la façon décrite précédemment à partir de l'image d'un document complet. La même méthode d'extraction est utilisée pour construire un graphe d'adjacence de régions $\mathcal{S}$ décrivant le symbole modèle dont on cherche les occurrences dans le document complet. Dès lors, la détection des occurrences du symbole associé à $\mathcal{S}$ au sein du plan associé au graphe $\mathcal{G}$ se ramène à un problème de recherche d'isomorphismes de sous-graphe. Cette recherche d'isomorphismes doit être tolérante aux erreurs dans le sens où, du fait du bruit présent sur les images ou résultant de l'extraction, l'étiquetage des nœuds et des arcs peut différer entre le graphe décrivant le symbole isolé et ses différentes occurrences au sein du graphe décrivant le document.

Le recherche d'isomorphisme de sous-graphe est un problème connu pour être NP-complet (Garey *et al.,* 1979). Un des algorithmes faisant référence en la matière est l'algorithme d'Ullman (Ullmann, 1976). Cet algorithme permet une exploration de la combinatoire tout en permettant une réduction significative de la complexité par un principe de retour arrière. Un certain nombre d'articles ont proposé des algorithmes avec des complexités moindres exploitant certaines contraintes (Hopcroft *et al.,* 1974, Cordella *et al.,* 1999) ou nécessitant un prétraitement de l'ensemble des graphes modèles (Messmer *et al.,* 1998). Certaines de ces approches sont des approches exactes. Les autres sont tolérantes aux erreurs. Les unes comme les autres imposent toutefois un étiquetage nominal des nœuds et des arcs du graphe. Leur utilisation pour traiter des graphes étiquetés par des attributs numériques est alors soumise à une étape préalable visant à transformer ces attributs numériques en attributs nominaux. Cette étape est généralement effectuée via une discrétisation ou une classification des attributs

numériques (Locteau, 2008). Cependant, une partie importante de l'information portée par les attributs numériques est alors perdue.

Nous n'avons pas connaissance d'algorithme permettant la recherche d'isomorphisme tolérant aux erreurs et apte à traiter directement des graphes où les nœuds et les arcs peuvent être étiquetés par des valeurs numériques vectorielles.

La section suivante présente la contribution principale de cet article, à savoir, un nouveau formalisme basé sur la programmation linéaire en nombres entiers pour la recherche d'isomorphismes de sous-graphe tolérante aux erreurs en présence de graphes étiquetés avec des valeurs numériques vectorielles, ce qui, à notre connaissance, présente une forte originalité.

3. Formulation 0-1 de l'isomorphisme de sous-graphe

3.1. *Programmation linéaire en nombres entiers*

Nous utilisons un programme linéaire en nombres entiers (Nemhauser *et al.*, 1988, Chinneck, n.d.). Un programme linéaire en nombres entiers se présente sous la forme générique suivante :

$$\min_x c^t x \tag{1a}$$

$$\text{sous la contrainte } Ax \leq b \tag{1b}$$

$$x \in C \subseteq \mathbb{Z}^n \tag{1c}$$

Dans cette formulation, $c \in \mathbb{R}^n, A \in \mathbb{R}^{n.m}, b \in \mathbb{R}^m$ sont les données du problème. Ce programme mathématique définit un ensemble de solutions pour le problème modélisé. La solution est un vecteur x de n variables, appartenant à $\mathbb{Z}$ dans le cas de la programmation en nombres entiers (1c). Les variables permettent d'exprimer des contraintes linéaires (1b). Une solution valide pour le problème est un vecteur x tel que les contraintes (1b) et (1c) sont respectées. Une telle solution est dite réalisable. Trouver une solution optimale consiste alors à minimiser la fonction objectif (1a) sur l'ensemble des solutions réalisables.

3.2. *Modélisation de la recherche d'isomorphismes de sous-graphe*

La modélisation que nous proposons n'utilise que des variables binaires. Elles sont de deux sortes et illustrées sur la figure 2 :

– pour chaque nœud $i \in V_{\mathcal{S}}$ et pour chaque nœud $k \in V_{\mathcal{G}}$, il existe une variable $x_{i,k}$ telle que $x_{i,k} = 1$ si les nœuds i et k sont mis en correspondance et $x_{i,k} = 0$ sinon.

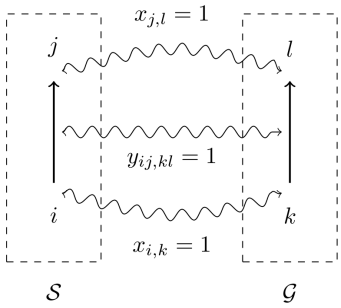


Figure 2. *Illustration de la mise en correspondance des nœuds et des arcs*

– pour chaque arc $ij \in E_\mathcal{S}$ et pour chaque arc $kl \in E_\mathcal{G}$, il existe une variable $y_{ij,kl}$ telle que $y_{ij,kl} = 1$ si les arcs ij et kl sont mis en correspondance et $y_{ij,kl} = 0$ sinon.

Étant donnés $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S})$ et $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$, supposons qu'on connaisse une distance $d_V : V_\mathcal{S} \times V_\mathcal{G} \to \mathbb{R}^+$ et une distance $d_E : E_\mathcal{S} \times E_\mathcal{G} \to \mathbb{R}^+$. Considérons deux nœuds $i \in V_\mathcal{S}$ et $k \in V_\mathcal{G}$. La mise en correspondance de i à k ($x_{i,k} = 1$) présente un coût de $d_V(i,k)$ alors que le fait de ne pas les associer ($x_{i,k} = 0$) présente un coût nul. Ce coût peut donc être noté $d_V(i,k) * x_{i,k}$. De même le coût d'association des arcs $ij \in E_\mathcal{S}$ et $kl \in E_\mathcal{G}$ peut être écrit $d_E(ij,kl) * y_{ij,kl}$. Il nous est alors possible d'écrire la fonction objectif qui vise à minimiser la somme des distances entre les éléments appariés :

$$\min_{x,y} \sum_{i \in V_\mathcal{S}} \sum_{k \in V_\mathcal{G}} d_V(i,k) * x_{i,k} + \sum_{ij \in E_\mathcal{S}} \sum_{kl \in E_\mathcal{G}} d_E(ij,kl) * y_{ij,kl} \quad [2]$$

Les contraintes du programme linéaire sont les suivantes :

– Chaque nœud de $\mathcal{S}$ doit être mis en correspondance avec un unique nœud de $\mathcal{G}$.

$$\sum_{k \in V_\mathcal{G}} x_{i,k} = 1 \quad \forall i \in V_\mathcal{S} \quad [3]$$

– Chaque arc de $\mathcal{S}$ doit être mis en correspondance avec un unique arc de $\mathcal{G}$.

$$\sum_{kl \in E_\mathcal{G}} y_{ij,kl} = 1 \quad \forall ij \in V_\mathcal{S} \quad [4]$$

– Chaque nœud de $\mathcal{G}$ doit être associé à au plus un nœud de $\mathcal{S}$.

$$\sum_{i \in V_\mathcal{S}} x_{i,k} \leq 1 \quad \forall k \in V_\mathcal{G} \quad [5]$$

– Si deux nœuds $i \in V_\mathcal{S}$ et $k \in V_\mathcal{G}$ sont mis en correspondance, un arc ayant i pour origine doit être mis en correspondance avec un arc ayant k pour origine.

$$\sum_{kl \in E_\mathcal{G}} y_{ij,kl} = x_{i,k} \quad \forall k \in V_\mathcal{G}, \forall ij \in E_\mathcal{S} \qquad [6]$$

– Si deux nœuds $j \in V_\mathcal{S}$ et $l \in V_\mathcal{G}$ sont mis en correspondance, un arc ayant j pour destination doit être mis en correspondance avec un arc ayant l pour destination.

$$\sum_{kl \in E_\mathcal{G}} y_{ij,kl} = x_{j,l} \quad \forall l \in V_\mathcal{G}, \forall ij \in E_\mathcal{S} \qquad [7]$$

– Enfin, nous précisons les contraintes indiquant que les variables du problème sont des variables binaires.

$$x_{i,k} \in \{0,1\} \quad \forall i \in V_\mathcal{S}, \forall k \in V_\mathcal{G} \qquad [8]$$

$$y_{ij,kl} \in \{0,1\} \quad \forall ij \in E_\mathcal{S}, \forall kl \in E_\mathcal{G} \qquad [9]$$

Les équations (2) à (9) forment le programme linéaire en nombres entiers utilisé pour résoudre la recherche d'isomorphisme. Toutes les solutions réalisables, celles qui satisfont les contraintes (3) à (9), correspondent toutes à un isomorphisme, cependant celle qui optimise la fonction objectif est celle pour laquelle le coût d'association est le plus faible.

Dès lors que la recherche d'isomorphisme est modélisée sous la forme d'un programme linéaire qu'il est possible de résoudre en utilisant un solveur mathématique. Dans cette étude, nous utilisons un solveur disponible sous licence CPL appelé SYMPHONY (Ralphs *et al.,* 2005). Dans la section suivante, nous présentons les expérimentations réalisées et discutons des résultats obtenus.

4. Évaluation

4.1. *Présentation des données*

Les données utilisées pour évaluer l'approche proposée sont extraites de la base `floorplans`[2]. Cette base est constituée de données synthétiques représentant différentes dispositions de symboles placés sur 10 fonds de plans architecturaux. Notre évaluation se base sur 200 images synthétiques de plans architecturaux correspondant aux 20 premières dispositions proposées pour chacun des fonds. Des exemples de ces images sont donnés sur la figure 3.

La tâche associée à cette base de données consiste à retrouver les occurrences des 16 symboles modèles présentés sur la figure 4.

2. `http ://mathieu.delalandre.free.fr/projects/sesyd/`

Figure 3. *Exemples d'images de la base* `floorplans` *correspondant à différents fonds de plan*

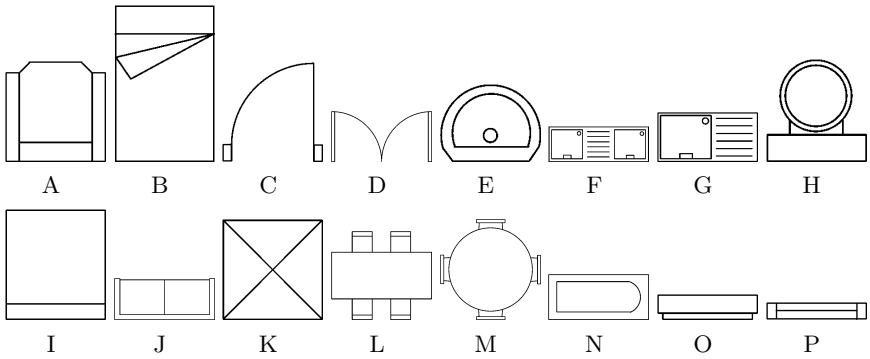


Figure 4. *Modèles des symboles recherchés*

Dans un premier temps, nous avons extrait la représentation structurelle de chacun des plans comme décrit en section 2. La figure 5 donne une vue du graphe d'adjacence de régions extrait pour un des plans de la base `floorplans`.

Grâce à une interface graphique développée pour l'occasion, il a été possible de constituer une vérité terrain pour la recherche d'isomorphisme de sous-graphe en identifiant au sein des 200 représentations structurelles les sous-graphes correspondant à des occurrences de symboles. Nous avons ainsi pu identifier que la base de plans contenait 5609 occurrences de symboles, soit environ 28 symboles par document en moyenne. Les sous-graphes correspondant

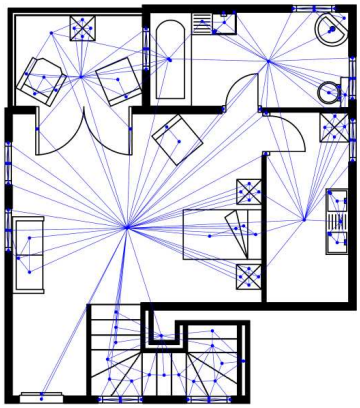


Figure 5. *Visualisation du graphe d'adjacence de région extrait sur un plan de la base* `floorplans`

symbole	nœuds	arcs	occurrences dans la base	documents contenant le symbole
A	4	10	248	106
B	4	10	300	160
C	2	0	840	200
D	2	0	100	60
E	3	4	112	112
F	7	16	150	130
G	4	8	208	151
H	4	6	208	188
I	2	4	434	176
J	3	6	255	192
K	4	10	1086	200
L	9	16	132	123
M	9	16	96	91
N	2	2	180	180
O	2	2	507	183
P	4	10	753	193

Tableau 1. *Répartition de la vérité terrain par type de symbole*

aux symboles contenaient en moyenne 4 nœuds et 7 arcs. En comparaison, les représentations structurelles des plans complets contiennent en moyenne 121 nœuds et 525 arcs.

4.2. *Mesures de performance*

À l'issue de la résolution par le solveur du programme linéaire en nombres entiers, nous disposons d'une proposition d'appariement entre le graphe représentant le symbole et celui représentant le plan en examinant lesquelles des variables x_{ik} et $y_{ij,kl}$ ont été mises à 1. Pour juger de la pertinence de cette proposition, il faut la comparer à la vérité terrain établie. Cette comparaison peut donner lieu à différents cas de figure :

– Dans le meilleur des cas, l'appariement proposé correspond exactement à celui de la vérité terrain (noté $=$)

– Certains des appariements proposés correspondent à la vérité terrain, alors que d'autres ne correspondent pas. (noté $\approx$)

– Aucun des appariements proposés ne correspond à la vérité terrain (noté $\neq$).

– Aucune proposition d'appariement n'est proposée. Ce cas de figure intervient lorsqu'aucune solution ne remplit les contraintes, c'est à dire qu'il n'existe pas de solution réalisable (noté $\emptyset$).

Dans le cas d'une proposition erronée d'isomorphisme, il convient de distinguer le cas où le symbole recherché est en réalité absent du document (noté $\neq |-$), auquel cas l'erreur est naturelle, du cas où le symbole est effectivement présent dans la vérité terrain (noté $\neq |+$).

Par ailleurs, si le solveur permet de retrouver la solution qui minimise la fonction objectif, il est possible de rechercher plusieurs occurrences en relançant une nouvelle recherche dans laquelle sera exclue la solution déjà trouvée, et ainsi de suite. Il est ainsi possible de rechercher plusieurs isomorphismes. Bien que d'autres configurations soient possibles, compte-tenu du contexte applicatif, lors de la recherche d'un nouvel isomorphisme, nous excluons des solutions possibles tout isomorphisme dans lequel apparaît un nœuds déjà apparié dans un isomorphisme précédent.

4.3. *Recherche d'une unique occurrence*

Dans une première expérimentation, nous avons recherché, dans chacun des plans, l'isomorphisme de coût minimal pour chacun des symboles modèles. Les résultats quantitatifs sont présentés dans le tableau 2. Globalement, ces résultats indiquent que sur les 3200 occurrences recherchées, 1612 correspondent exactement à des symboles présents sur les plans, 380 correspondent partiellement (au moins 1 nœud apparié à bon escient et au moins un nœud apparié à mauvais escient). 755 recherches ont été effectuées alors que le symbole n'était pas présent dans le plan. Finalement, le système n'a produit de véritables erreurs ($\neq |+$) que dans 453 des 2445 cas où un symbole pouvait être trouvé. On remarque également que les deux dernières colonnes du tableau indiquent que

Symbole	$=$	$\approx$	$\neq \mid +$	$\neq \mid -$	$\emptyset$
A	88	3	15	94	0
B	48	72	40	40	0
C	56	72	72	0	0
D	0	11	49	140	0
E	112	0	0	88	0
F	96	34	0	70	0
G	51	21	79	49	0
H	40	128	20	12	0
I	114	0	62	24	0
J	170	2	20	8	0
K	154	30	16	0	0
L	123	0	0	77	0
M	85	6	0	109	0
N	179	0	1	20	0
O	124	0	59	17	0
P	172	1	20	7	0
Total	1612	380	413	755	0

Tableau 2. *Résultats de la recherche d'une occurrence de symbole par plan*

le système a toujours proposé une solution réalisable lors de la recherche d'une unique occurrence.

Le tableau 2 révèle également des disparités entre des classes proposant de bons résultats (par exemple les classes E, M et N) et d'autres pour lesquelles les résultats sont très faibles (classes C, D et G). Deux explications peuvent être données. Concernant les classes C et D, la représentation en graphe d'adjacence de régions n'est pas adaptée à ces symboles qui ne présentent que deux régions blanches non adjacentes (cf. Fig. 4C et 4D). Le graphe correspondant est donc restreint à deux nœuds non reliés par un arc. De ce fait, il n'existe pas de contrainte topologique pour l'appariement ce qui conduit à de nombreuses erreurs. Concernant la classe G, on peut observer que le symbole correspondant est complètement inclus dans celui de la classe F, ce qui, là aussi, est source de confusions.

4.4. *Recherche de plusieurs occurrences*

Étant donnés les résultats obtenus lors de la recherche d'une unique occurrence et considérant le fait qu'un même symbole est susceptible d'apparaître à plusieurs reprises sur un même document, nous avons souhaité évaluer la recherche de plusieurs isomorphismes. Compte-tenu du fait qu'une composante connexe ne peut appartenir qu'à un seul symbole, nous avons, dans cette ex-

Symbole	$=$	$\approx$	$\neq \mid +$	$\neq \mid -$	$\emptyset$
A	210	7	328	2090	7365
B	157	131	166	2367	7179
C	158	669	1882	6849	442
D	19	61	1092	8386	442
E	112	0	0	5325	4563
F	104	46	0	892	8958
G	98	110	126	2642	7024
H	40	168	32	3615	6145
I	397	6	1368	6784	1445
J	233	2	503	2935	6327
K	473	427	987	1063	7050
L	132	0	0	954	8914
M	90	6	0	1080	8824
N	180	0	18	8438	1364
O	467	0	1899	6257	1377
P	648	3	311	1706	7332
Total	3518	1636	8712	60383	84751

Tableau 3. *Résultats de la recherche de 50 occurrences de symbole par plan*

périmentation, paramétré la recherche de telle sorte que soit exclu des solutions réalisables tout isomorphisme faisant apparaître un nœud déjà apparié dans un isomorphisme précédent.

Le tableau 3 présente les résultats d'une recherche de 50 occurrences de chaque symbole modèle pour chacun des 200 plans de la base `floorplans`. Dans ce tableau, les colonnes $\neq \mid -$ et $\emptyset$ indiquent respectivement le nombre d'erreurs et de recherches infructueuses pour cause d'absence de solution réalisable sachant que toutes les occurrences réellement présentes ont été trouvées précédemment. Il ne s'agit donc pas à proprement parler d'erreur. Même s'il persiste, comme dans le cas de la recherche d'une unique occurrence, des disparités entre classes, globalement, on retrouve exactement 3518 (62,7%) et partiellement 1636 (29,2%) des 5609 occurrences de symboles réellement présentes dans les documents. En estimant qu'une correspondance partielle suffit à considérer que l'occurrence du symbole est détectée, on atteint un rappel de 92%.

Ces bons résultats doivent cependant être tempérés. En effet, un nombre non négligeable de fausses détections sont opérées avant d'avoir retrouvé la dernière occurrence réelle du symbole cherché (colonne $\neq \mid +$). Parallèlement, en l'état actuel, notre système n'est pas en mesure de déterminer le nombre de symboles de chaque type présents sur un plan. De ce fait, dans cette expérimentation, nous avons volontairement choisi une valeur importante du nombre de détections à proposer au regard du nombre de symboles réellement présents

Symbole	rappel	précision
A	0,88	0,08
B	0,96	0,10
C	0,98	0,09
D	0,80	0,01
E	1,00	0,02
F	1,00	0,14
G	1,00	0,07
H	1,00	0,05
I	0,93	0,05
J	0,92	0,06
K	0,83	0,30
L	1,00	0,12
M	1,00	0,08
N	1,00	0,02
O	0,92	0,05
P	0,86	0,24
global	0,92	0,07

Tableau 4. *Valeur du rappel et de la précision par classe pour une recherche de 50 occurrences*

(recherche de 50 occurrences par type de symbole pour chaque plan) sans que cela soit suffisant pour les retrouver tous (rappel < 1). Mais, *a contrario*, cela a conduit le système à proposer des occurrences possibles alors que toutes les occurrences réelles avaient déjà été trouvées (colonne $\neq |-$). Enfin, la recherche s'est dans certains cas interrompue d'elle-même lorsqu'il n'existait plus de solution réalisable (colonne $\emptyset$). Compte-tenu de ce nombre important de fausses détections, la recherche de 50 occurrences de chaque symbole pour chaque plan a aboutit à un rappel de 92% pour une précision de 7%. Nos travaux futurs s'emploieront à proposer des compromis précision/rappel permettant de proposer des modes de fonctionnement intermédiaires en adaptant le nombre d'occurrences cherchées.

5. Conclusion

Dans cet article, nous présentons une approche pour la recherche d'isomorphisme exact de sous-graphe tolérant aux erreurs basée sur une programmation linéaire en nombres entiers. Cette approche est implantée grâce l'utilisation d'un solveur libre et appliquée à la recherche d'occurrences de symbole dans des documents architecturaux en utilisant des représentations à partir de graphe pour décrire aussi bien le plan que les symboles à chercher.

Les expérimentations réalisées sur des bases publiques montrent des performances intéressantes. En effet, lors de la recherche d'une occurrence de symbole, si celui-ci est effectivement présent, une de ses occurrences est retrouvée au moins partiellement dans plus de 80% des cas. De même, lors de recherche de multiples occurrences, les symboles sont localisés dans plus de 60% des cas exactement et dans presque 30% des cas partiellement.

Cependant, ce taux de rappel est obtenu au détriment d'une précision relativement faible. En effet, notre système ne permet pas en l'état de déterminer automatiquement le nombre d'occurrences à localiser, et si certaines occurrences sont trouvées tardivement, de nombreuses fausses détections sont expliquées par le fait bon nombre de recherches sont lancées sans qu'il n'y ait plus de nouveaux symboles à retrouver.

Nous envisageons de mener une étude qui permettrait de palier cette carence. Pour ce faire, nous projetons d'établir une étude statistique sur la valeur des coûts des isomorphismes (valeur de la fonction objectif). De cette façon, nous pourrions pour chaque classe de symbole établir des valeurs seuils de la fonction objectif qui permettraient éventuellement de prendre des décisions de rejet des isomorphismes associés à un coût supérieur.

À plus long terme, nous envisageons également de mener une étude sur l'impact et le paramétrage des fonctions de distance (par exemple une pondération des différentes caractéristiques) employées dans le calcul de la fonction objectif.

Remerciements

Les travaux décrits dans cet article ont partiellement été financés dans le cadre du projet NAVIDOMASS ANR-06-MDCA-12 par l'Agence National pour la Recherche.

6. Bibliographie

Bay H., Ess A., Tuytelaars T., Gool L. V., « SURF : Speeded Up Robust Features », *Computer Vision and Image Understanding*, vol. 110, n° 3, p. 346-359, 2008.

Chang C.-J. C. F., Lu C.-J., « A linear-time component-labeling algorithm using contour tracing technique », *Computer Vision and Image Understanding*, vol. 93, p. 206-220, 2004.

Chinneck J. W., « Practical optimization. A gentle introduction », `http ://www.sce.carleton.ca/faculty/chinneck/po.html`, n.d.

Cordella L. P., Foggia P., Sansone C., Vento M., « Performance evaluation of the VF graph matching algorithm », *Proccedings of the International Conference on Image Analysis and Processing*, p. 1172-1177, 1999.

Delalandre M., Pridmore T., Valveny E., Locteau H., Trupin E., « Building synthetic graphical documents for performance evaluation », *Graphics Recognition. Recent*

Advances and New Opportunities, vol. 5046 of *Lecture Notes in Computer Science*, p. 288-298, 2008.

di Baja G. S., Thiel E., « Skeltonization algorithm running on path-based distance maps », *Image and Vision Computing*, vol. 14, p. 47-57, 1996.

Garey M. R., Johnson D. S., *Computers and Intractability : A Guide to the Theory of NP-Completeness*, Freeman & co., 1979.

Hopcroft J., Wong J., « Linear time algorithm for isomorphism of planar graphs », *Proceedings of the sixth annual ACM Symp. Theory of Computing*, p. 172-184, 1974.

Lazbenik S., Schmid C., Ponce J., « Semi-Local Affine Parts for Object Recognition », *Proceedings of the British Machine Vision Conference*, p. 779-788, 2004.

Lladós J., Martí E., Villanueva J. J., « Symbol Recognition by Error-Tolerant Subgraph Matching between Region Adjacency Graphs », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, n° 10, p. 1137-1143, 2001.

Locteau H., Modélisation et acquisition de connaissances pour l'indexation/interprétation de documents hétérogènes, PhD thesis, Université de Rouen, 2008.

Lowe D. G., « Object Recognition form Local Scale-Invariant Features », *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, p. 1150-1157, 1999.

Messmer B. T., Bunke H., « A New Algorithm for Error-Tolerant Subgraph Isomorphism Detection », *IEEE Transactions on Pattern Analysis and Machine Inteligence*, vol. 20, n° 5, p. 493-504, 1998.

Nemhauser G. L., Wolsey L. A., *Integer and combinatorial optimization*, Wiley-Interscience, New York, NY, USA, 1988.

Queshri R. J., Ramel J.-Y., Cardot H., « De l'appariement de graphes symboliques à l'appariements de graphes numériques : Application à la reconnaissance de symboles », *Actes de la Conférence Internationale Francophone sur l'Écrit et le Document*, p. 31-36, 2006.

Ralphs T. K., Güzelsoy M., *The Next Wave in Computing, Optimization, and Decision Technologies*, vol. 29 of *Operations Research/Computer Science Interfaces Series*, Springer US, chapter The Symphony Callable Library for Mixed Integer Programming, p. 61-76, 2005.

Rusiñol M., Lladós J., Sánchez G., « Symbol Spotting in Vectorized Technical Drawings Throug a Lookup Table of Region Strings », *Pattern Analysis and Applications*, 2009.

Teague M., « Image analysis via the general theory of moments », *Journal of the Optical Society of America*, vol. 70, n° 8, p. 920-930, 1980.

Terrades O. R., Tabbone S., Valveny E., « A review of shape descriptors for document analysis », *Proceedings of the ninth International Conference on Document Analysis and Recognition*, p. 227-231, 2007.

Ullmann J. R., « An Algorithm for Subgraph Isomorphism », *Journal of the ACM*, vol. 23, n° 1, p. 31-42, 1976.